Geometria Computacional

MAC 0331

Pedro Gigeck Freire 18/03/2020

NUSP: 10737136

Lista 1

② [CLRS 33.4-1] O prof Maqui Sperto teve uma ideia genial e veio com um novo esquema para que o algoritmo encontre o par mais próximo verificando, no Combine, somente a distância entre cada ponto p nos pontos da faixa, que estão no vetor f, e os 6 pontos que estão a seguir de p em f. A ideia é sempre colocar os pontos da reta separadora no conjunto E da esquerda. Assim, não haverá um par de pontos coincidentes sobre a reta com um ponto em E e outro em D. Portanto, no máximo 7 pontos podem estar no retângulo $d \times 2d$. Onde está a bobagem do esquema proposto pelo professor Sperto?

A bobagem está na redução das instâncias da recursão.

Não conseguimos garantir que cada chamada recursiva processe menos elementos se os pontos em cima da linha forem pra esquerda. Se tivermos mais de 3 pontos em cima da linha, a recursão nunca chegará no caso base, os pontos ficarão infinitamente indo pra esquerda.

Então a bobagem é que algoritmo pode nunca terminar.

Além disso, não conseguimos garantir a complexidade do algoritmo $O(n \lg n)$ pois os tamanhos das recorrências poderão resultar num pior caso (como no Quick Sort) $O(n^2)$

④ Modifique a fase de Combinar do Algoritmo visto em aula para que seja calculada a distância entre cada ponto da faixa e apenas pontos do outro lado da partição feita em Divida. Faça a modificação de modo a manter o consumo de tempo em $O(n \lg n)$

Mudaremos, primeiramente, a função Candidatos para que ela retorne os candidatos de cada metade em vetores separados.

```
Candidatos (X, a, p, r, d)
    q ← ⌊(p+r)/2⌋
    Tesq ← 0
```

```
    t_DIR ← 0
    para k ← p até r faça
        se |X[a[k]] − X[q]| < d
            então se a[k] ≤ q
                    então t_esq ← t_esq + 1
                          f_esq[t_esq] ← a[k]
                    senão t_DIR ← t_DIR + 1
                          f_DIR[t_DIR] ← a[k]
    devolva (f_esq, t_esq, f_DIR, t_dir)
```

Agora, adaptamos o Combine para processar as duas metades (Na outra página eu tento explicar como ele funciona) (Recomendo olhar as duas páginas uma embaixo da outra pra ver o código completo)

```
Combine(X, Y, a, p, r, d_E, d_D)
    d ← min{d_E, d_D}
    (f_esq, t_esq, f_dir, t_DIR) ← Candidatos(X, a, p, r, d)
    menor_esq ← 1
    menor_DIR ← 1
    para i ← 1 até t_esq − 1 faça
        enquanto menor_DIR < t_DIR e Y[f_DIR[menor_DIR]] < Y[f_esq[i]] faça
            menor_DIR ← menor_DIR + 1
        j ← menor_DIR
        enquanto j < t_DIR e Y[f_DIR[j]] − Y[f_esq[i]] < d faça
            d' ← Dist(X[f_DIR[j]], Y[f_DIR[j]], X[f_esq[i]], Y[f_esq[i]])
            se d' < d
                então d ← d'
            j ← j + 1
    para i ← 1 até t_DIR − 1 faça
        enquanto menor_esq < t_esq e Y[f_esq[menor_esq]] < Y[f_DIR[i]] faça
            menor_esq ← menor_esq + 1
```

```
17          j ← menor_esq
18          enquanto j < T_esq e Y[f_esq[j]] − Y[f_DIR[i]] < d faça
19                d' ← Dist(X[f_esq[j]], Y[f_esq[j]], X[f_dir[i]], Y[f_DIR[i]])
20                se d' < d
21                     então d ← d'
22                j ← j + 1
23    devolva d
```

O algoritmo repete o processamento para as duas metades.

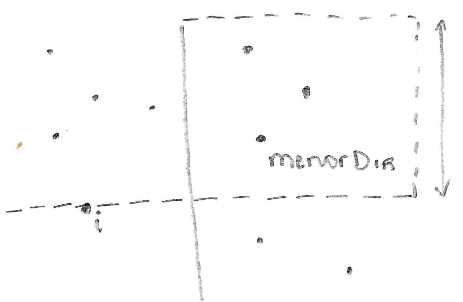
Da linha 5 a 13 percorremos os pontos da esquerda

Da linha 14 a 22 percorremos os pontos da direita

Em cada metade, olhamos para um ponto i e temos que descobrir quem são os pontos da outra metade que estão acima de i, então os índices <u>$menor_{DIR}$ e $menor_{ESQ}$ guardam o primeiro ponto da outra metade com Y maior que i</u>

Então olhamos os pontos j da outra metade que estão acima do ponto i e a uma distancia vertical no máximo d.

Por exemplo



Então, para o ponto i, medimos a distância de i aos pontos j desse quadrado da direita

Como $T_{esq} + T_{DIR} - 2$ é o total de pontos candidatos e as linhas 6-7 e 15-16 executam, no máximo, considerando todas as iterações, $T_{esq} + T_{DIR}$ vezes (já que $menor_{esq}$ e $menor_{DIR}$ nunca diminuem) o consumo de tempo do Combine continuará linear, então o algoritmo continua $O(n \lg n)$.